Kinesthetic Teaching of End-Effector and Null-Space Motion Primitives with Learned Task Priorities

handed in
MASTER'S THESIS

M.Sc Jiabin Wu

born on the Aug.24th,1988
living in:
Felsennelkenanger 19/N225
80937 Munich
Tel.: 0176 4564 0883

Chair of
AUTOMATIC CONTROL ENGINEERING
Technical University of Munich

Univ.-Prof. Dr.-Ing./Univ. Tokio Martin Buss
Univ.-Prof. Dr.-Ing. Dirk Wollherr

Supervisor:	M.Sc Matteo Saveriano, Prof. Ph.D Dongheui Lee
Start:	Mar.15th, 2016
Intermediate Report:	July 19th, 2016
Delivery:	Dec.9th, 2016

TECHNISCHE UNIVERSITÄT MÜNCHEN

LEHRSTUHL FÜR STEUERUNGS- UND REGELUNGSTECHNIK

ORDINARIUS: UNIV.-PROF. DR.-ING./UNIV. TOKIO MARTIN BUSS

March 2016

M A S T E R   T H E S I S
for
Jiabin Wu
Mat.-Nr. 03657284, field of study EI

Kinesthetic Teaching of End-Effector and Null-Space Motion Primitives with Learned Task Priorities

Problem description:

Kinesthetic teaching, i.e. the manual guidance of the robot towards the task execution, is a natural and intuitive way to teach new skills [1]. When the robot is redundant, one should consider how the redundant degrees-of-freedom (DoF) can be used in a fruitful manner. In [2], an approach is proposed for kinesthetic teaching of null-space motion primitives. The approach in [2] considers fixed task priorities during the execution, i.e., the user has to pre-define end-effector and null-space tasks.

In this Master Thesis work the student has to overcome the limitations of the work in [2] by considering variable task priorities. Variable task priorities will be learned by optimizing a suitable cost function [3]. The learning approach has also to take into account eventual constraints in the task demonstrations. To this end, the student has to implement an approach to automatically extract constraints from the given demonstrations. These constraints will be explicitly considered in the algorithm which learns the variable task priorities.

Tasks:

- Literature overview on variable impedance control and learning task priorities.
- Implementation of an approach to learn variable task priorities.
- Evaluation on a KUKA LWR IV+ 7 DoF manipulator and a NAO humanoid robot.

Bibliography:

[1] D. Lee and C. Ott. Incremental Kinesthetic Teaching of Motion Primitives Using the Motion Refinement Tube, in *Autonomous Robots*, 2011.

[2] M. Saveriano, S. An and D. Lee. Incremental Kinesthetic Teaching of End-Effector and Null-Space Motion Primitives, in *International Conference on Robotics and Automation*, 2015.

[3] V. Modugno, G. Neumann, E. Rueckert, G. Oriolo, J. Peters and S. Ivaldi. Learning soft task priorities for control of redundant robots, in *Int. Conf. on Robotics and Automation*, 2016.

[4] S. Calinon, F. Guenter and A. Billard. On Learning, Representing, and Generalizing a Task in a Humanoid Robot, in *Transactions on Systems, Man, and Cybernetics, Part B*, 2007.

Supervisor:	M.S. Matteo Saveriano
Start:	15.03.2016
Intermediate Report:	19.07.2016
Delivery:	09.12.2016

(D. Lee)
Univ.-Professor

Abstract

In planning and control of a redundant robot, a key problem is how to satisfy multiple tasks and constraints in a low cost way. This work extends the work of [SAL15] by considering variable task priorities in order to overcome the limitations. In this work, we formulate an approach that calculates the optimal way for the execution of multiple tasks, while taking the constraints into consideration. An Implementation of (1+1)CMA-ES algorithm, a stochastic optimization procedure, is applied to learn the task priories. The (1+1)CMA-ES will be intruced in the first place. Then a method for learning is proposed based on [MNR$^+$16]. Finally the proposed approach will be implemented in *KUKA LWR IV+7 DoF* robot and *aldebaran NAO* robot.

Acknowledgement

Firstly, I would like to express my sincere gratitude to my advisor, PhD candidate M.Sc. Matteo Saveriano, for the continuous support of my master study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.
I would like to thank the leader of this Chair, Prof. Ph.D Dongheui Lee, for her insightful comments and encouragement, but also for the hard questions which incented me to widen my work to a new level.
Many thanks also go to Dr. Dirk Arnold, leading author of [AH10], for offering me the source code of (1+1)CMAES algorithm and patient explanation.
Last but not the least, I want to thank all the colleges in this Lab, for giving me a friendly and cosy surroundings, which benefits my work a lot.

Jiabin Wu

Contents

Chapter 1

Introduction

Kinesthetic teaching, i.e. the manual guidance of the robot towards the task execution, is a natural and intuitive way to teach new skills [SAL15]. In [SAL15] an approach is proposed for kinesthetic teaching of end-effector and null-space motion primitives. This approach considers fixed task priorities during the execution, i.e. the user has to pre-defined the priorities of the tasks with a hand tuned threshold, which limits the applicability of the method. On the other side, task with the highest priority will be accurately executed, however the execution of the remaining tasks would not be guaranteed.
In this work, instead of fixed priorities, approach with dynamical priorities is proposed. In this approach, each given task is assigned a priority vector, which is acquired by a learning procedure though an stochastic optimization. In this way, all the tasks can be executed in the optimal mode. Another major contribute of this work is, we propose a constraint-handling mechanism which will, in case of a constraint contradiction, prevent the approach from crash, and drive the approach to satisfy all the constraints as well as possible.

1.1 Related Work

Many efforts have been made in this field. Concerning the aspect of optimization algorithm, The CMAES (Covariance Matrix Adaptation Evolution Strategy) is one of the most powerful evolutionary algorithms for real-valued optimization with many successful applications[IHR07].The main advantages of the CMAES lie in its invariance properties, which are achieved by carefully designed variation and selection operators, and in its efficient (self-) adaptation of the mutation distribution.
Besides, CMAES is a stochastic, or randomized, method for real-parameter (continuous domain) optimization of non-linear, non-convex functions. A normal way to optimize an object function in continuous domain is using the gradients, which requires that the object function must be derivable. If the object function was unknown (e.g. black box) or the gradients were not available, a substitute method

need to be applied to solve the problem. The CMA-ES algorithm is proven to be a powerful tool under this circumstance. It is independent from the structure of the object function and requires only the input-output mapping of the function. This also eases the users when modifying the object function.
The (1+1)CMAES introduced by Igel et al.[ISH06] is a variant of the CMAES that combines the well known (1+1) selection scheme with the covariance matrix adaption. It reduces the computational effort from $O(n^3)$ to $O(n^2)$ [ISH06]; However, both CMAES and (1+1)CMAES are unconstrained optimization, can not handle constraint problem. A variant version of (1+1)CMAES that can handle constraints is introduced by Arnold et al.[AH10]. The key idea underlying this constraint handling approach is to reduce the variances of the distribution of offspring candidate solutions in the direction of the normal vectors of constraint boundaries in the vicinity of the current parental solution.
We decided to use the algorithm of [AH10] as our optimization algorithm.
Concerning the aspect of prioritized tasks control in redundant manipulator, many controllers have been developed for this purpose. Generally there are two ways to prioritize the tasks: static prioritization and dynamic prioritization. In the static prioritization, the tasks are divided into different levels based on its importance, critical tasks will be assigned higher priorities. In the execution, tasks with lower priorities will be executed in the null space of the higher-priority tasks [SRK+13]. the work in [DLH09] introduces an optimization framework called prioritized optimization control, in which a nested sequence of objectives are optimized so as not to conflict with higher-priority objectives. In the dynamic prioritization, all the tasks are combined though weights [SPB11]. By tuning the weights in the optimization process, the performance of the whole system could be optimized. As in the work of [ODAS15], a new approach for dynamic control of redundant manipulators is proposed to deal with multiple prioritized tasks at the same time by utilizing null space projection techniques. The compliance control law is based on a new representation of the dynamics wherein specific null space velocity coordinates are introduced.
In our work, we will combine the (1+1)CMAES algorithm with the dynamic prioritization method.

1.2 Task Statement

The main problems come from different aspects:

- Literature overview on variable impedance control and learning task priorities;
- Implementation of the CMA-ES algorithm to learn variable task priorities;
- Constraints handling;

- Experimental evaluation on a KUKA LWR IV+7 DoF manipulator and Aldebaran NAO.

The rest of this report is organized as follows. Chapter 2 introduce the technical details of our proposed approach, which includes a recap of [SAL15], (1+1)CMAES and the Optimized Task Control. Then in the chapter 3 are experiments and results. Finally comes the conclusion.

Chapter 2

Technical Approach

This chapter will present our proposed approach in details. As mentioned above, this work is an improvement and extension of work [SAL15], so a short recap of previous work and its limitations are given at first to emphasize again the motivation of our work. After the recap is the overview of the proposed approach. In section 2.2, the method for priorities learning is introduced, which is adapted from [MNR$^+$16]. Then in section 2.3 we present a new multi-task control that handles constrained tasks.

2.1 Overview of the proposed approach

In this section, we present the key idea of developed approach for multiple tasks handling. Before introducing our new approach, it is necessary to recap some key points of the previous work in [SAL15].

2.1.1 Recap of previous work

The main contribute of [SAL15] consists in combining incremental learning algorithms with a customized multi-priority kinematic controller,the so-called Task Transition Control (TTC), which guarantees a smooth human-robot interaction.
In the TTC, $[T_i, T_j]$ $(i \neq j)$ represents a prioritized task set, in which T_i has higher priority and the order of the tasks can be switched by a hand-tuned force f_e. Task with higher priority will be executed in the task space, and the task with lower priority will be executed in the nullspace of the higher one. Given are the force threshold f_s and two different tasks: 1) end-effector task T_{ee} and 2) interaction control task T_{ic} .
Case 1, T_{ic} is an end-effector task. If $f_e < f_s$, then the task stack is $[T_{ee}, T_{ic}]$, which means that $[T_{ee}]$ will be executed and $[T_{ic}]$ will be ignored; Else if $f_e > f_s$, then $[T_{ic}]$ will be executed and $[T_{ee}]$ will be ignored.
Case 2, T_{ic} is a non-end-effector task, elbow task for example. If $f_e < f_s$, then the task stack is $[T_{ee}, T_{ic}]$, which means that $[T_{ee}]$ will be accurately executed and $[T_{ic}]$

will only be executed in the null-space of T_{ee}; Else if $f_e > f_s$, then $[T_{ic}]$ will be accurately executed and the execution of $[T_{ee}]$ will be set to second place.
The advantages of this approach are obvious, however there are also so drawbacks. 1) It is difficult for the robot to estimate the external force because of the random touching points; while on the user side, the subjective perception of the force differs among users. 2) In the above-mentioned case 2, the execution of task with higher priority has minimized error ($||T_{desired} - T_{measured}||$), while the execution error of the other task depends on whether there is a conflict in task set and how horrible the conflict is. In order to cope with these drawbacks, we propose a new approach, in which all the tasks can be executed as best as possible.

2.1.2 Key ideas of proposed approach

In contrast to TTC, we propose OTC (Optimized Task Control). OTC works with robotic kinematics instead of robotic dynamics in TTC. So torque condition is removed. In OTC, each task will be assign a priority, which will be optimized by CMAES algorithm during the process, so that all tasks could be executed as best as possible. Besides, TTC can also handle tasks with constraints. By configuring the constraints, it is possible to modify the results to meet different needs.

2.2 Priorities learning

In this section, a method for learning the task priorities is proposed, which is based on the work of [MNR+16]. Please refer to this paper for more details.

2.2.1 Basic idea

The figure 2.1 shows the overview of the method. The controller θ consists of a weighted combination of sub-controller, where the weights represent the priorities. A learning loop enable the online optimization of the weights.

- step 1, predefine all the elementary tasks: $task_1$ $task_2$ $...task_n$, and initialize all the weights/priorities randomly;
- step 2, calculate the controller $\vec{\theta_i}$ for each task;
- step 3, calculate the overall controller $\vec{\theta}$ from the weighted $\vec{\theta_i}$;
- step 4, send $\vec{\theta}$ to the robot and measure the parameter set $\vec{\phi}$, which depends on the requirement from the cost function;
- step 5, evaluate the cost optimize the weights, then go to step 3;

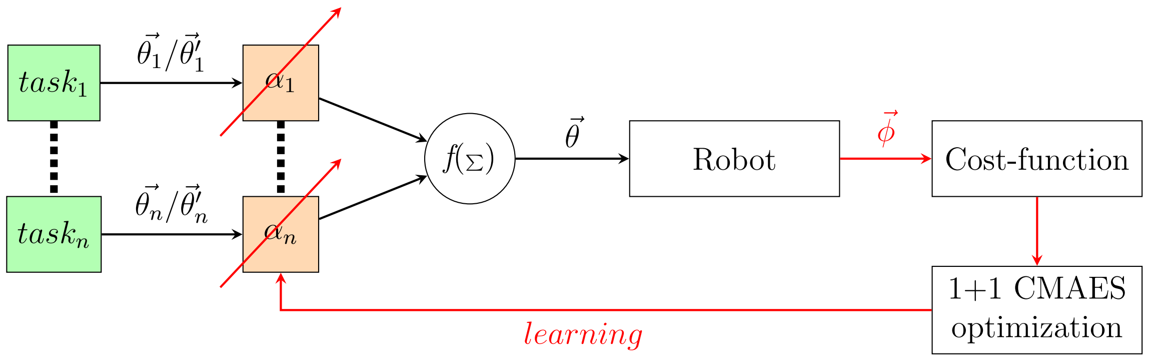


Figure 2.1: overview of proposed method (figure adapted from [MNR+16])

Now, assume that all the θ_i are available, then the overall controller can be calculated.

$$\vec{\theta}(t) = \vec{\theta}(t-1) + dt\left(\sum_{i=1}^{n} \alpha_i \vec{\theta}'_i(t-1)\right) \tag{2.1}$$

where
α_i is the priority of θ_i and depends on π_i;
π_i is the **parameter that needs to be learned** from CMA-ES.
Further explanation for equation 2.1 is that the priorities are modelled by a weighted sum of normalized Radial Basis Functions:

$$\alpha_i(\pi_i, t) = S\left(\frac{\sum_{k=1}^{n_r} \pi_{ik}\Psi(\mu_k, \sigma_k, t)}{\sum_{k=1}^{n_r} \Psi(\mu_k, \sigma_k, t)}\right) \tag{2.2}$$

where
$\Psi(\mu_k, \sigma_k, t) = exp(-1/2[(t-\mu_k)/\sigma)_k]^2)$ is the set of basic function, with (μ_k, σ_k) being the mean and variance;
n_r is the number of the basic function;
$\pi_i = (\pi_{i1}, ...\pi_{in_r})$ is the set of parameters of each task priority,which **corresponds to $\vec{x}$ in equation 2.3** ;
S(.) is a sigmoid function which squashes its output to the range of [0 1];

2.2.2 1+1 CMAES

As mention before, (1+1)CMAES algorithm can reach the global optimum with a relative lower computational effort, while considering the task constraints. So we decided to use the algorithm of [AH10] as our optimization algorithm.

Algorithm of the (1+1)CMAES in [AH10]

1, Generate offspring candidate solution y according to:

$$y = x + \sigma A z \tag{2.3}$$

where, x is the parental candidate solution, z is standard normally distribution components, σ is the global step size and A is the Cholesky factor of C ($C = AA^T$).

2, For $j = 1, 2, ..., m$ determine whether $g_j(y) > 0$ (the j_{th} constraint is violated) and update v_j if $g_j(y) > 0$ according to:

$$v_j \leftarrow (1 - c_c)v_j + c_c A z \tag{2.4}$$

where, v_j is the constraint vector and $c_c \in (0, 1)$ determines how quickly the information present in the constraint vectors fades.

3, If y is infeasible (at least one constraint is violated in step 2), then compute $w_j = A^{-1}v_j$ for all $j = 1, 2, ..., m$ and update the transformation matrix A according to:

$$A \leftarrow A - \frac{\beta}{\sum_{j=1}^{m} \mathbf{1}_{g_j(y)>0}} \frac{v_j w_j^T}{w_j^T w_j} \tag{2.5}$$

where, $w_j = A^{-1}v_j$ and $\mathbf{1}_{g_j(y)>0}$ equals one if $g_i(y) > 0$ and zero otherwise.
Then this iteration is completed.

4, If y is feasible (no constraint is violated in step 2), then evaluate the cost function $f(y)$ and update the success probability estimate according to:

$$P_{succ} \leftarrow (1 - c_P)P_{succ} + c_P \mathbf{1}_{f(y) \leq f(x)} \tag{2.6}$$

Then update the global step size according to:

$$\sigma \leftarrow \sigma \; exp\left(\frac{1}{d} \frac{P_{succ} - P_{target}}{1 - P_{target}}\right) \tag{2.7}$$

where, P_{succ} is the success probability estimate and c_P is a constant within (0,1).

5, if $f(y) \leq f(x)$, then replace x and y, update search path according to:

$$s \leftarrow (1 - c)s + \sqrt{c(2 - c)} A z \tag{2.8}$$

and update matrix A according to:

$$A \leftarrow \sqrt{1 - c_{cov}^{+}} A + \frac{\sqrt{1 - c_{cov}^{+}}}{\parallel w \parallel^2} \left(\sqrt{1 + \frac{c_{cov}^{+} \parallel w \parallel^2}{1 - c_{cov}^{+}}} - 1 \right) s w^T \tag{2.9}$$

Then this iteration is completed.

6, otherwise if $f(y) > f(x)$, and $f(y)$ is inferior to its fifth order ancestor, update matrix matrix according to:

$$A \leftarrow \sqrt{1+c_{cov}^{-}}A + \frac{\sqrt{1+c_{cov}^{-}}}{\parallel z \parallel^2}\left(\sqrt{1-\frac{c_{cov}^{-}\parallel z \parallel^2}{1-c_{cov}^{+}}}-1\right)zz^T \tag{2.10}$$

In Matlab, there are already some existed optimization functions which can also handle constraints, e.g. $fmincon$, however, compare to (1+1)CMAES, they have some limitations or disadvantages:
1, the constraint format is fixed, all the constraints must be written in matrix form, which is unrealistic for most of the situations.
2, when dealing with the some problem, matlab functions consume far more time than CMAES.

2.3 Optimized Task Control

In our approach, we have designed a new task controller, namely Optimized Task Control (OTC). Instead of working on Dynamics, where the system inputs are torques, we work on the Kinemastics, where the system inputs are configurations (joint angles in our case). So the hand-tuning force condition is removed and all the tasks will be treated equally.
Given are a set of tasks $T_1T_2...T_j...T_m$ with corresponding controllers $q_1q_2...q_j...q_m$. Each task is assigned an priority that is dynamic and will be optimized during the optimization. Then an optimized controller q_{op} is generated, which would guarantee all the tasks can be executed as best as possible.
For easy understanding, following parts consider here only the two tasks case T_1, T_2. Multi-tasks cases can be deduced in the similar way.

2.3.1 Calculation of the optimal controller

For each time step i, calculate the optimal controller as follows[SSVO10]:

$$q_{op}(i) = q_{op}(i-1) + \dot{q}_{op}(i)d_t \tag{2.11a}$$
$$\dot{q}_{op}(i) = \alpha_1(i)[J_1^+(i) + P_1 J_2^+(i)\dot{v}_2(i)] + \alpha_2(i)[J_2^+(i) + P_2 J_1^+(i)\dot{v}_1(i)] \tag{2.11b}$$
$$P_1 = I_n - J_1^+ J_1 \tag{2.11c}$$
$$P_2 = I_n - J_2^+ J_2 \tag{2.11d}$$
$$J_1^+ = J_1^T(J_1 J_1^T)^{-1} \tag{2.11e}$$
$$J_2^+ = J_2^T(J_2 J_2^T)^{-1} \tag{2.11f}$$

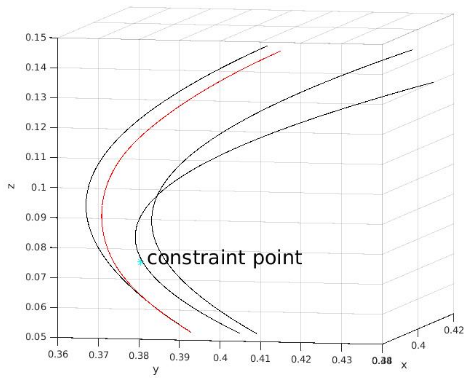
where,
d_t is the sampling period;
J_m^+ is the pseudo inverse of $J_m(m = 1, 2)$;
α_m is the priority of T_m, which will be optimized by CMAES and must satisfy the $Barycentric\ Coordinates$ condition:$\sum_{j=1}^{m} \alpha_j = 1$[AL15];
P_m is the projector in the null space of J_m;
I_n is the identity matrix;
$\dot{v}_m$ is linear velocity of T_m;

2.3.2 Constraint handling

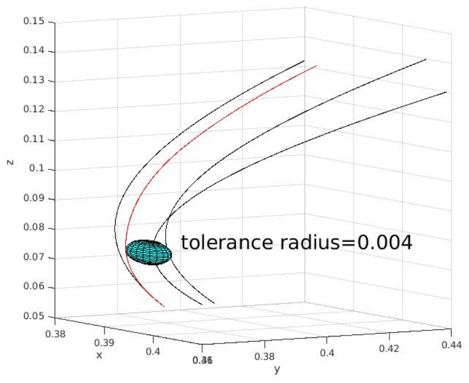
Constraint handling is an important way to balance performances among tasks. With higher requirements comes stricter constraint handling. The disadvantage of the is, in case of constraint contradictions that are difficult to be predicted in most of the cases, the running time is infinitive. To cope this defect and reduce the risk, we divide the constraints into two levels:
1, A-level constraints. These constraints corresponding to the constraints mentioned in step 2 of the algorithm in 2.2.2, which will be handled first. Only after all the A-level constraints are satisfied, can the algorithm move on;
2, B-level constraints. These constraints will be set as $costfunction$,corresponding to the cost function f mentioned in step 4 of the algorithm in 2.2.2. The OTC will drive the solution to satisfy these constraints as well as possible after A-level constraints are satisfied.

2.3.3 Constraint extraction

Some of the tasks will be generated from trajectory cluster by Kinesthetic teaching. The coach will guide the robot to move a few times, then the robot will moves itself tracking the guided trajectory. The teaching trajectory cluster consists of a few similar trajectories. A smooth trajectory would be generate with GMM/GMR model. Then the robot will track the new generated trajectory. Since the robot will deviate from the guided trajectory during the optimization process, so it is necessary to



(a) Feature point extraction



(b) acceptance region

Figure 2.2: Feature point and its acceptance region

extract some features as constraints, only when the robot satisfies these constraints, could be considered as finished the teaching task.

In the experiment parts, teaching happens in tests in KUKA. As in figure 2.2 (a), each teaching task consists of 3 guiding (black lines). A smooth trajectory (red line) is generated from the cluster. In the extraction of the feature, the 3 teaching trajectories will firstly be clustered into 10 datasets by $k - means$ algorithm, and the center point of the dataset with lowest covariance is selected as feature point (cyan star). It is unrealistic to drive the robot to go though exactly the feature point because of unacceptable time consumption in the stochastic optimization procedure. So as long as the robot gets close enough to the feature point, could be considered as satisfying the constraint. An acceptance region is shown as a sphere with radius=0.004 in figure 2.2 (b) .

Chapter 3

Experiments and results

the proposed approach will be evaluated in two robot models: $KUKALWRIV + 7DoFmanipulator$ and $AldebaranNAOv3.3$.

3.1 Evaluation on KUKA

Two kinds of tasks are to be considered: End-effector task and Elbow task. Models for the End-effector and Elbow are built in the first place, then OTC of different modes are tested.

3.1.1 Environment setting

Running environment: MATLAB;
Indispensable toolbox: Robotics toolbox for MATLAB. [Cor11]

LWR kinematic model

DH parameters: see table 3.1

Base transformation matrix (transformation matrix from first-joint frame to the base frame):

$$M\_base = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.11 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Tool transformation matrix (transformation matrix from end-effector frame to the 7th joint frame):

$$M\_tool = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.18 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Joint	*theta*	*d*	*a*	α	*jointtype*
1	0	0.2	0	$\pi/2$	revolute
2	0	0	0	$-\pi/2$	revolute
3	0	0.4	0	$-\pi/2$	revolute
4	0	0	0	$\pi/2$	revolute
5	0	0.39	0	$\pi/2$	revolute
6	0	0	0	$-\pi/2$	revolute
7	0	0.078	0	0	revolute

Table 3.1: DH-parameters for the KUKA Endeffector

Then generate the model with *SerialLink* function in Robotics toolbox for MATLAB: figure3.1

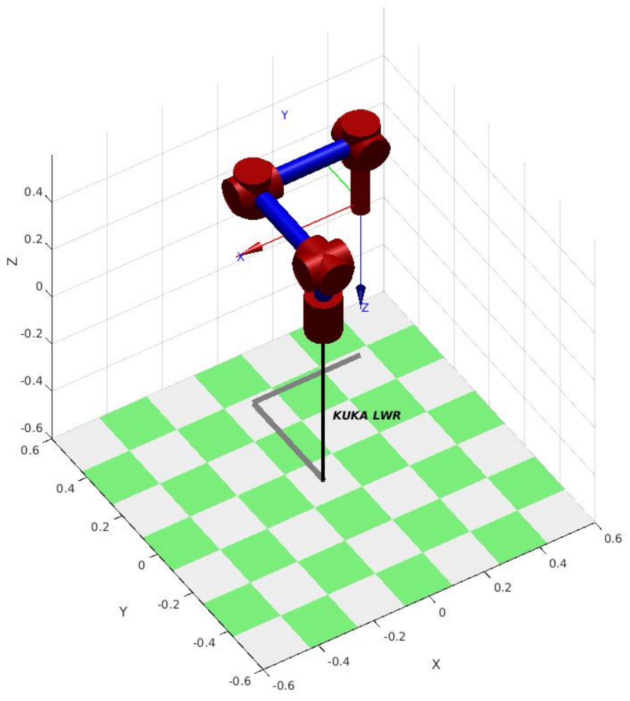


Figure 3.1: KUKA End-effector Model

Elbow model

DH parameters: see table 3.2
Base transformation matrix:

Joint	*theta*	*d*	*a*	α	*jointtype*
1	0	0.2	0	$\pi/2$	revolute
2	0	0	0	-$\pi/2$	revolute
3	0	0.4	0	-$\pi/2$	revolute

Table 3.2: DH-parameters for the KUKA Elbow

$$M\_base = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.11 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Tool transformation matrix (transformation matrix from end-effector frame to the 3rd joint frame):

$$M\_tool = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Then generate the model with *SerialLink* function in Robotics toolbox for MATLAB: figure3.2

3.1.2 Experimental results

Task definition

Task one: end-effector tracks the given/teaching trajectory;
Task two: elbow tracks the given/teaching trajectory;
Teaching: happens at either elbow or end-effector;
Parameter to be minimized: tracking error (sum up error of each step).

Colour mapping in the following trajectory figures:
Blue: original original trajectories should be tracked;
Black: teaching trajectories;
Red: new trajectory to be tracked, which is generated from the black lines though GMM/GMR;
Green: final trajectories after optimization process.

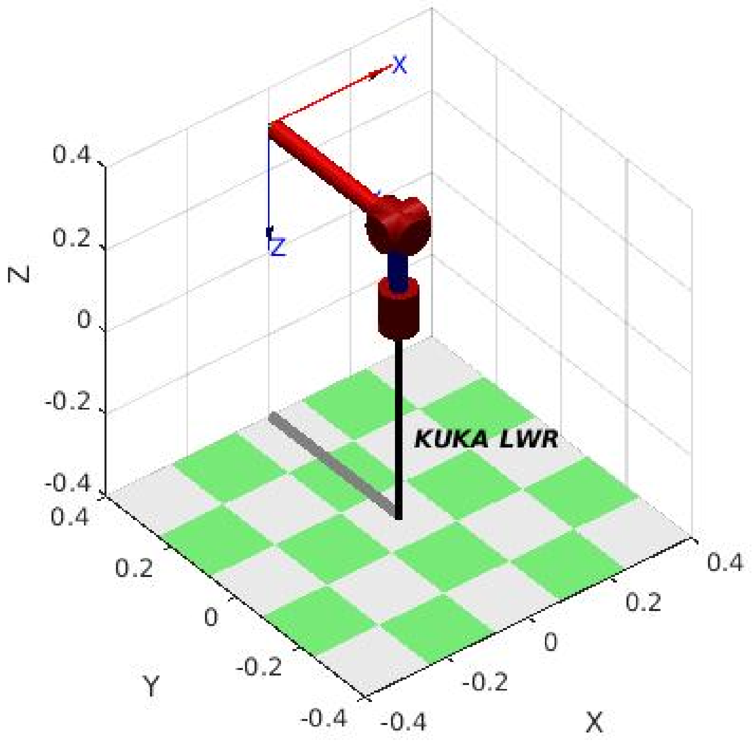


Figure 3.2: KUKA Elbow Model

Simulate TTC

In order to highlight the relevance and differences between OTC and the TTC, the first experiment is to configure OTC to simulate TTC.
Configuration:
1, given are one elbow trajectory and one end-effector trajectory;
2, teaching at elbow;
3, set the tracking error of elbow as B-level constraint, no A-level constraint.

Whis configuration the OTC works similarly to $Case2 \ \ f_e \ > \ f_s$ in 2.1.1;
Result 1:
Tracking error: end-effector 4.3391 against elbow 0.0057;
See Figure 3.3 and 3.4

Analysis 1:
The blue lines are the original trajectories should be tracked. Then teaching happens at elbow, shown as 3 black lines in the figure3.4. The red line is the new trajectory to be tracked, and the end-effector keeps on tracking the original one. The green lines in both figures are the final trajectories. Since only elbow tracking error is

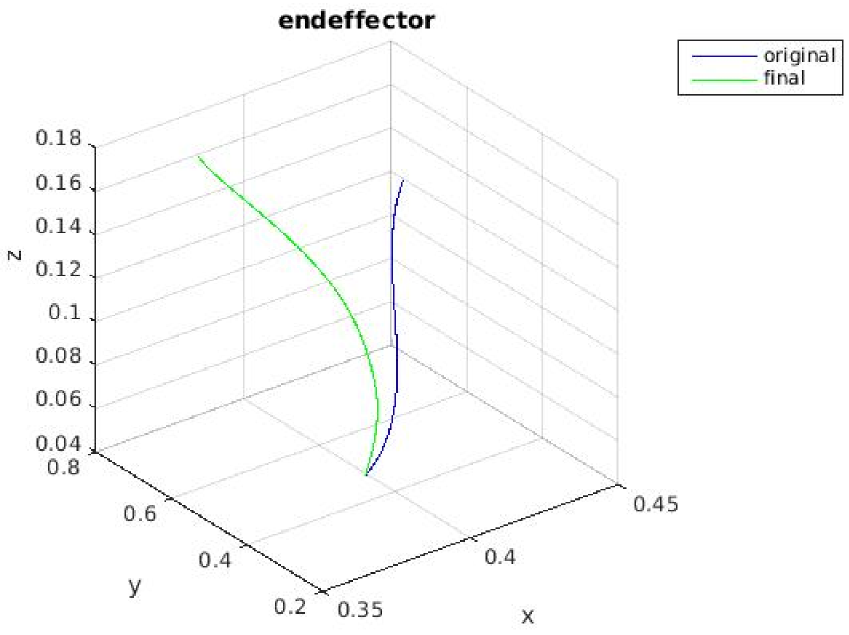


Figure 3.3: End-eddector performance in TTC case 2

considered, which means that elbow task has higher priority, the elbow tracks the red line quite well (red line and green line overlap), while in figure 3.3 green line and blue line diverges from each other obviously.

When modify the configuration as follows:

1, given are one elbow trajectory and one end-effector trajectory;
2, teaching at end-effector;
3, set the tracking error of end-effector as B-level constraint, no A-level constraint.

then the OTC works similarly to $Case1 \;\; f_e \; > \; f_s$ in 2.1.1;
Result 2:
Tracking error: end-effector 0.0092 against elbow 1.5261;
See Figure 3.5 and 3.6

Optimization on both tasks without difference

In this part of tests, OTC will execute both tasks as best as possible. However because of task contradiction, neither could reach the optimum.
Configuration:
1, given are one elbow trajectory and one end-effector trajectory;

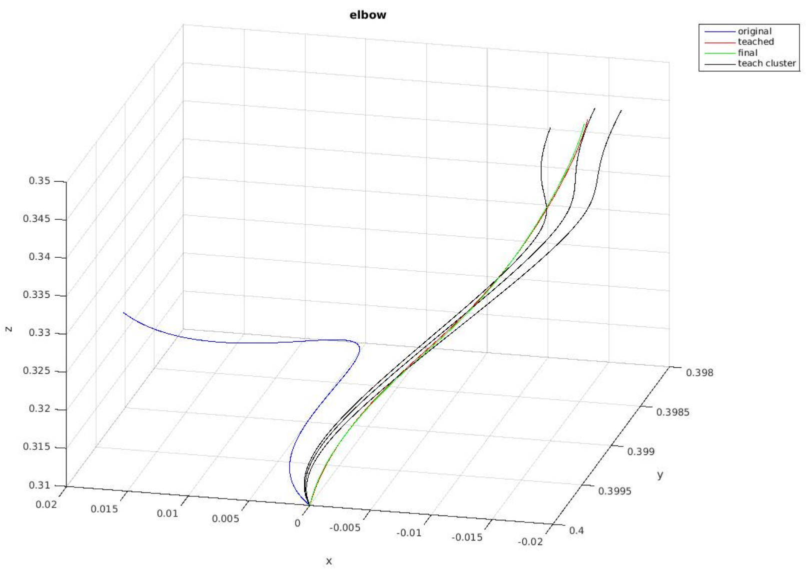


Figure 3.4: Elbow performance in TTC case 2

2, teaching at elbow;
3, set the tracking error of both end-effector and elbow as B-level constraint, no A-level constraint.

Result 3:
Tracking error: end-effector 1.1065 against elbow 1.1063;
See figure 3.7 3.8 and 3.9; In figure 3.9, omega1 is end-effector task priority, omega2 is elbow.

Analysis 3:
The optimization process minimize the tracking errors of both tasks equally, so compromising point is found to balance the errors between corresponding worst and best cases in result 1 and result 2 of part **Simulate TTC**;

Similarly, change the teaching to end-effector, get the similar results.
Result 4:
Tracking error: end-effector 0.9058 against elbow 0.9058;
See figure 3.10 3.11 and 3.12;

Optimization on both tasks with difference

In this part, constraint will be firstly extracted from the teaching cluster, and set as A-level constraints. The error of the task without teaching is set as B-level con-

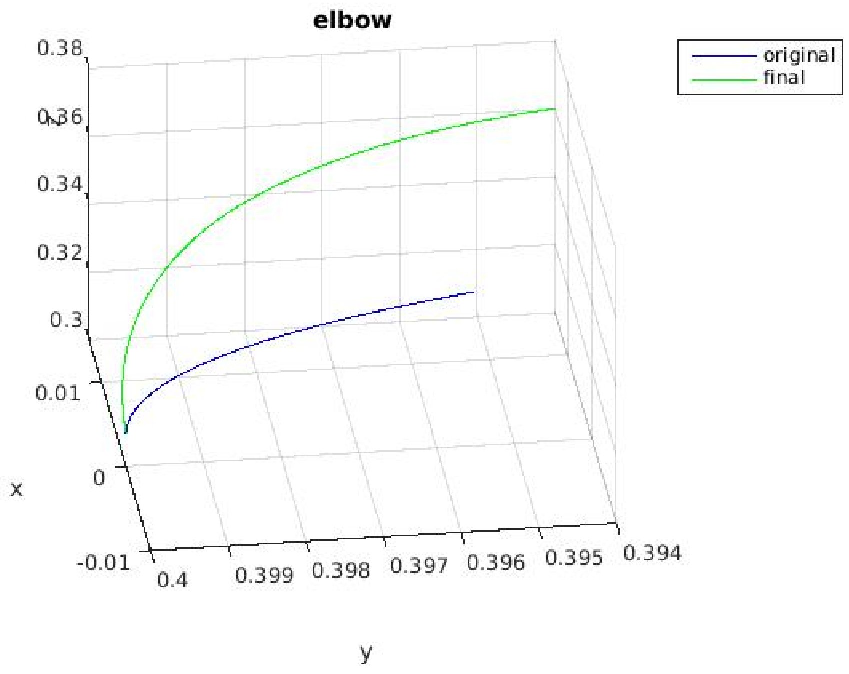


Figure 3.5: Elbow performance in TTC case 1

straint.

Configuration:
1, given are one elbow trajectory and one end-effector trajectory;
2, teaching at elbow;
3, extract one constraint point from the teaching cluster via 2.3.3 and set as A-level constraint. The final elbow trajectory must be close enough to this point (distance lower than 0.002);
4, set the tracking error of end-effector as B-level constraint.

Result 5:
Tracking error: end-effector 3.1728 against elbow 0.4545;
See figure 3.13 3.14 and 3.15;

Analysis 5:
As shown in the figure 3.15, the OTC will execute majorly the elbow task first. It drives the elbow trajectory to get close enough to the constraint point which is is illustrated as cyan star in figure 3.14. Then at about 65th step, OTC starts driving the end-effector to the given trajectory.

Configuration:
1, given are one elbow trajectory and one end-effector trajectory;

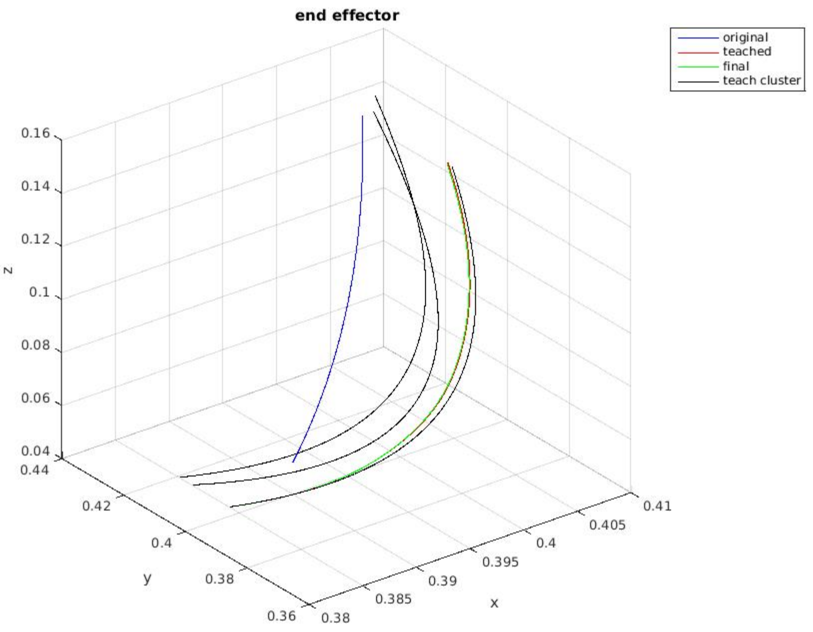


Figure 3.6: End-eddector performance in TTC case 1

2, teaching at end-effector;
3, extract one constraint point from the teaching cluster, and set as A-level constraint. The final end-effector trajectory must be close enough to this point (distance lower than 0.004);
4, set the tracking error of elbow as B-level constraint.

Result 6:
Tracking error: end-effector 1.7633 against elbow 0.31625;
See figure 3.16 3.17 and 3.18;

Analysis 6:
A-level is satisfied at about 19th step, and TTC starting driving the elbow to given trajectory.

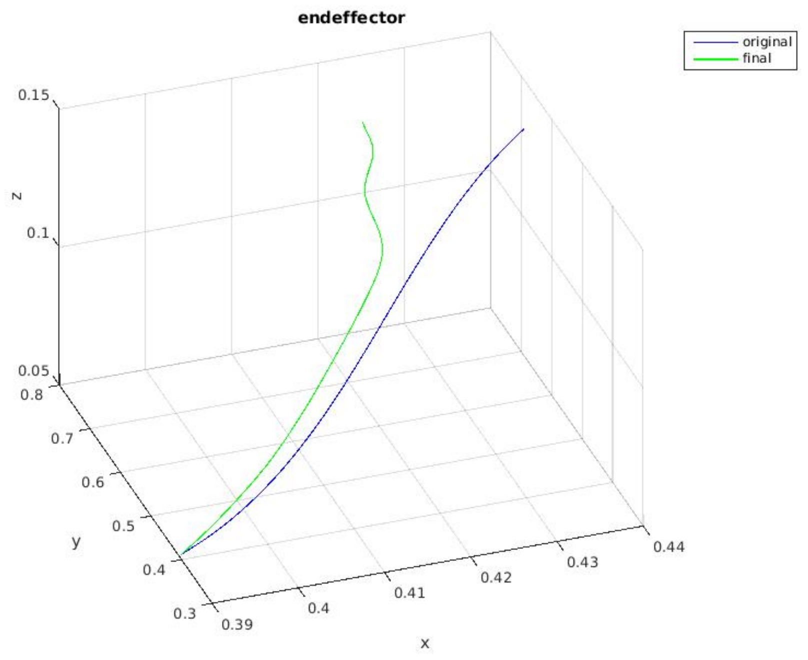


Figure 3.7: End-effector performance by optimization on both tasks without difference

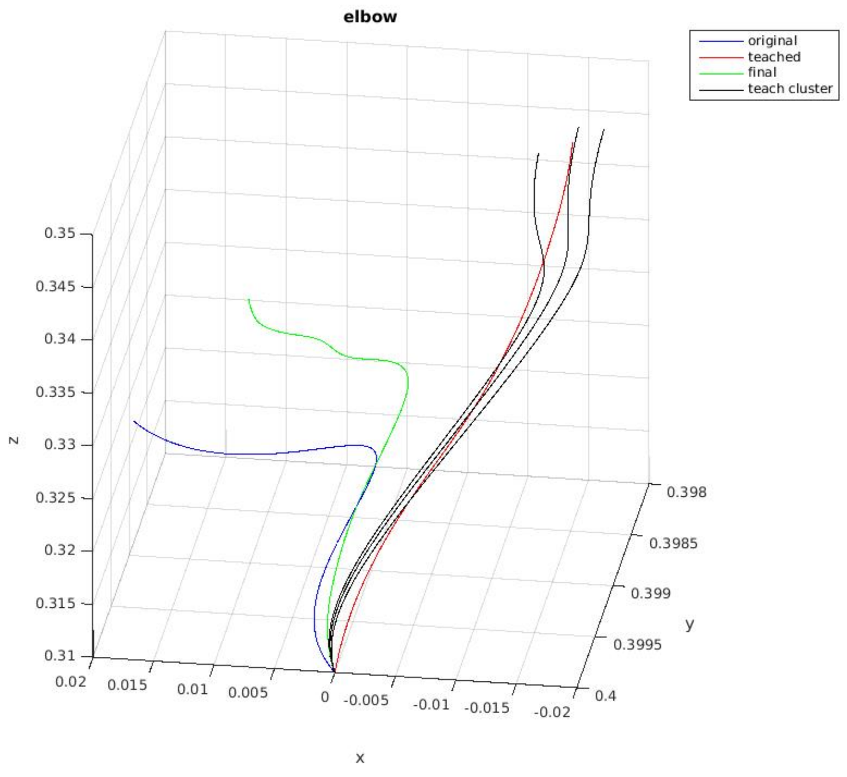


Figure 3.8: Elbow performance by optimization on both tasks without difference

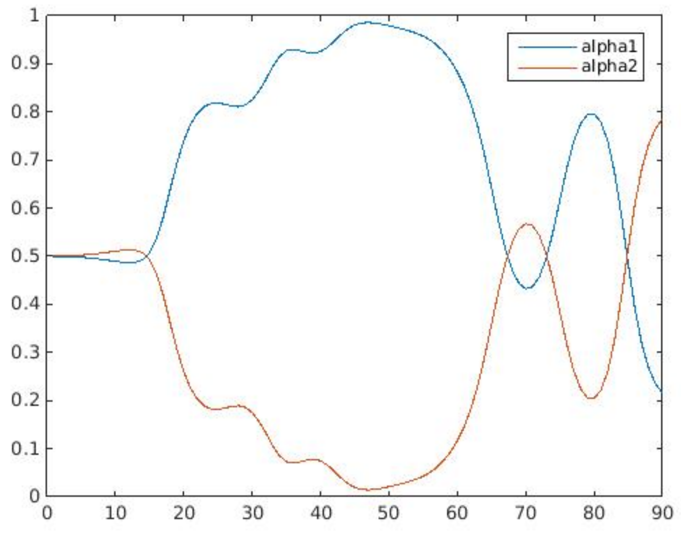


Figure 3.9: Priorities by optimization on both tasks without difference

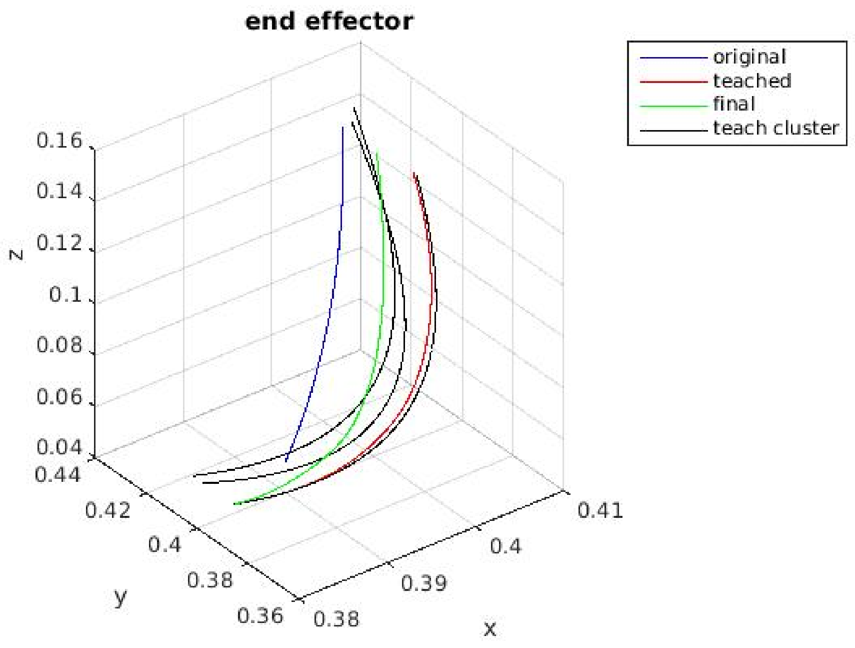


Figure 3.10: End-effector performance by optimization on both tasks without difference

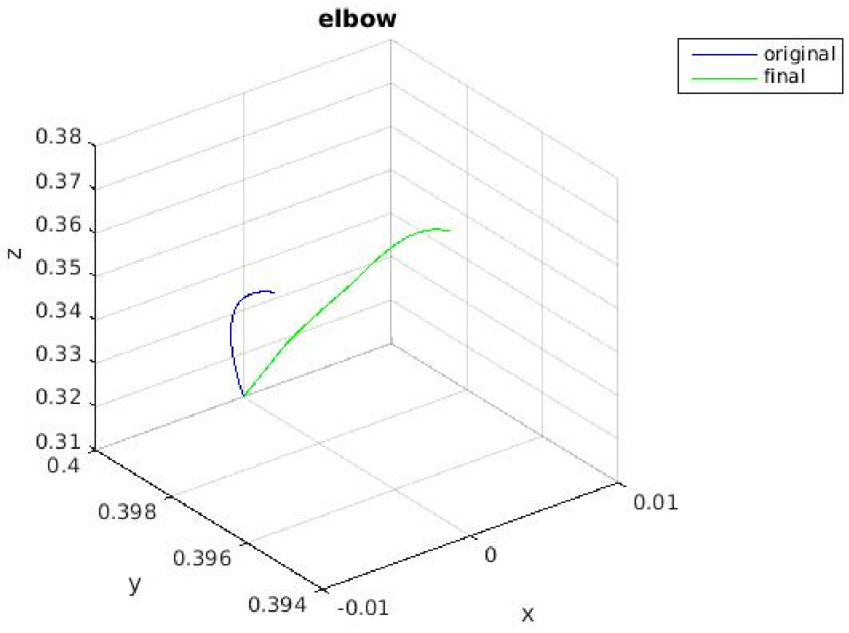


Figure 3.11: Elbow performance by optimization on both tasks without difference

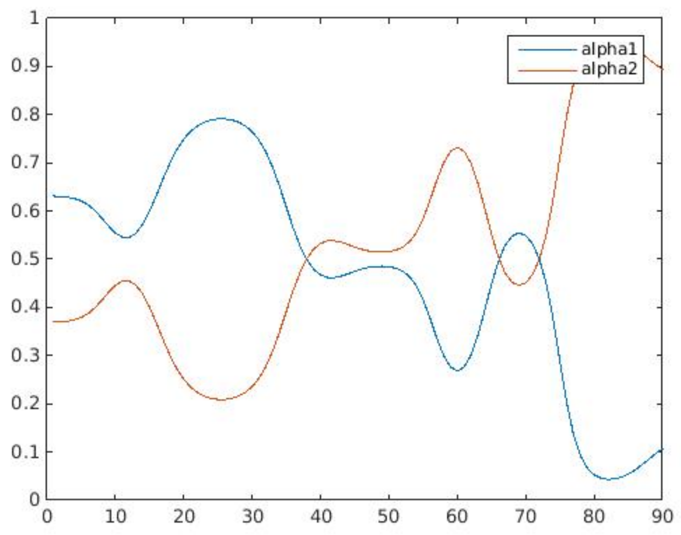


Figure 3.12: Priorities by optimization on both tasks without difference

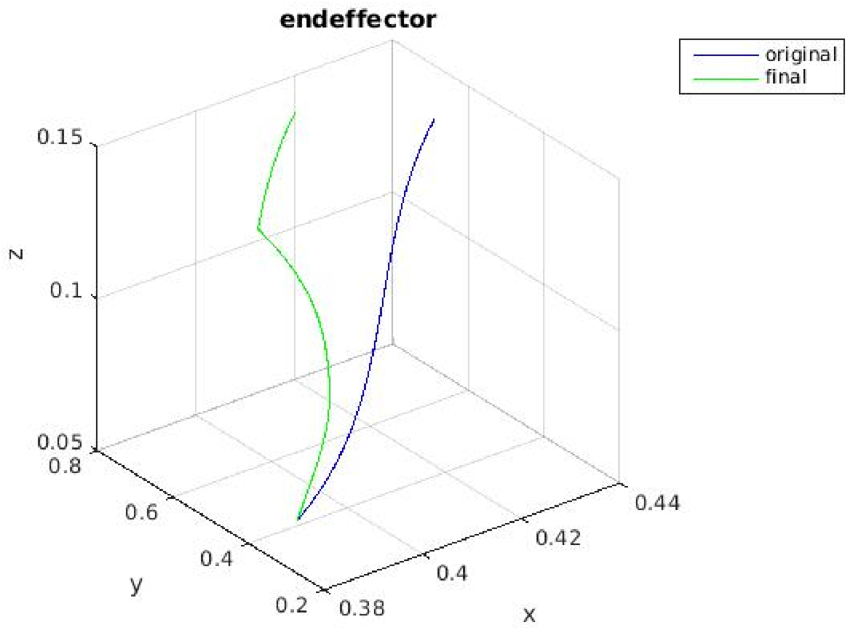


Figure 3.13: End-effector performance by optimization on both tasks with difference

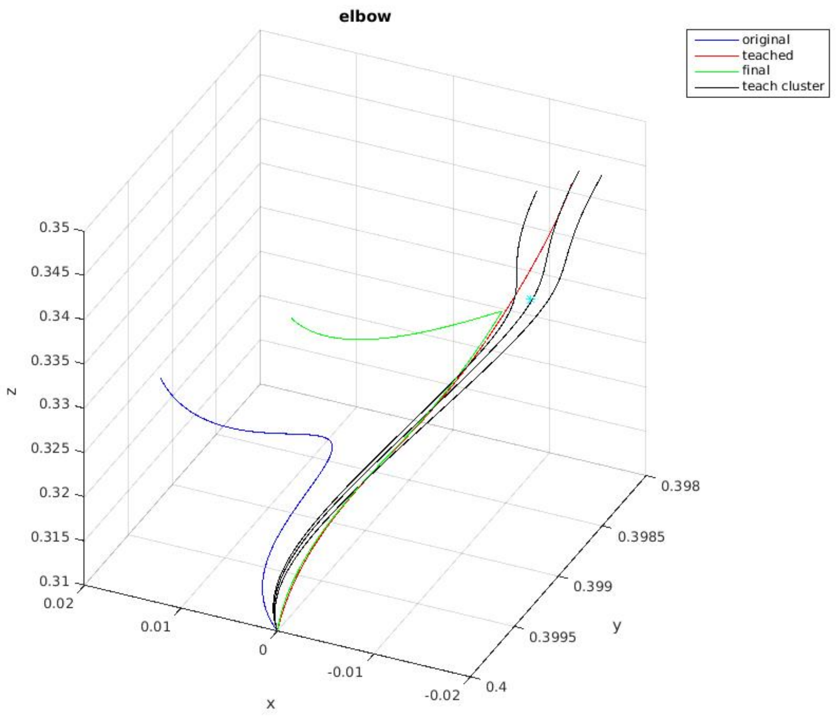


Figure 3.14: Elbow performance by optimization on both tasks with difference

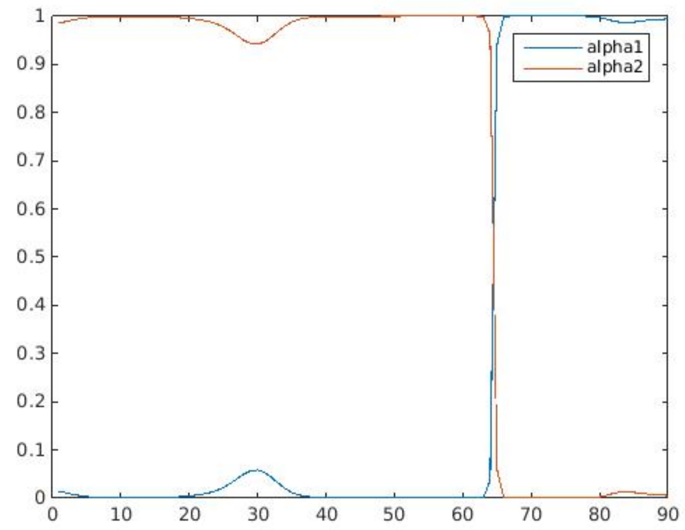


Figure 3.15: Priorities by optimization on both tasks with difference

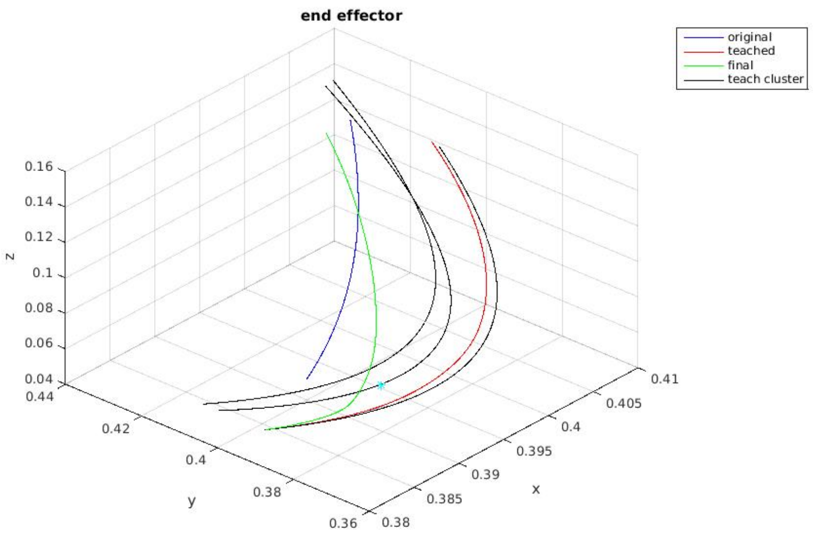


Figure 3.16: End-effector performance by optimization on both tasks with difference

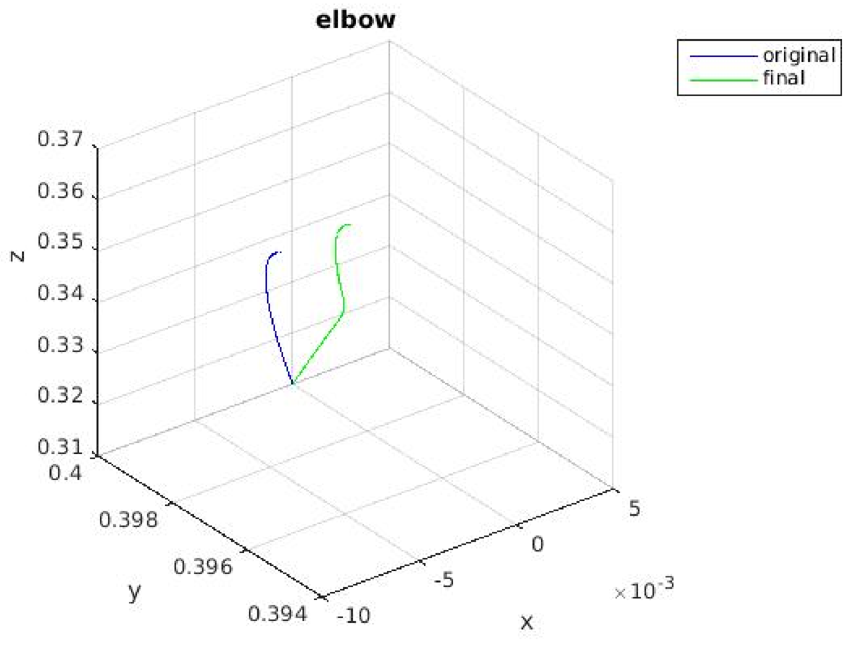


Figure 3.17: Elbow performance by optimization on both tasks with difference

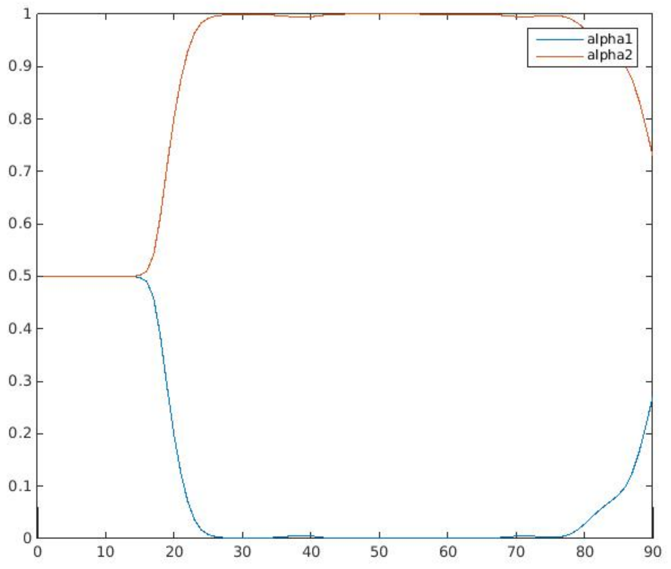


Figure 3.18: Priorities by optimization on both tasks with difference

3.2 Evaluation on NAO

Two kinds of tasks are to be considered: left-arm task and right-arm task.The approach is only tested on the arms. Since two arm chains are not connected, one arm chain could not affect the other without rotating the leg joints. however the model would be too complicated if the legs are involed. So two more joints are added to isolate the legs and connect the arms. One joint for the torso to turn left or right; one for bending over.

3.2.1 Environment setting

Running environment: MATLAB;
Indispensable toolbox: Robotics toolbox for MATLAB. [Cor11]

NAO left-arm model

DH parameters: see table 3.3

Joint	*theta*	*d*	*a*	α	*jointtype*
1	0	0	0	-$\pi/2$	revolute
2	0	0.1	0	$\pi/2$	revolute
3	0	0.098	0	-$\pi/2$	revolute
4	$\pi/2$	0	0	0	revolute
5	0	0.105	0.015	$\pi/2$	revolute
6	0	0	0	-$\pi/2$	revolute

Table 3.3: DH-parameters for the NAO left-arm

Base transformation matrix (transformation matrix from first-joint (torso tuning joint) frame to the base frame):

$$M\_base = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Tool transformation matrix (transformation matrix from end-effector frame to the 6th-joint (LWristYaw,[Kof12]) frame):

$$M\_tool = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & -0.1137 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Then generate the model with *SerialLink* function in Robotics toolbox for MATLAB: figure3.19

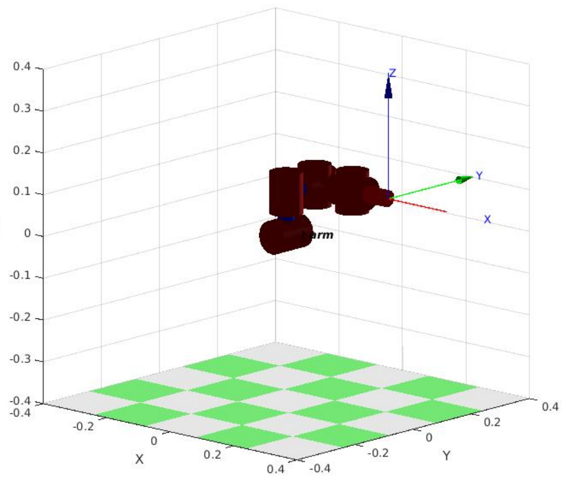


Figure 3.19: NAO left-arm Model

NAO right-arm model

DH parameters: see table 3.4

Joint	*theta*	*d*	*a*	α	*jointtype*
1	0	0	0	-$\pi/2$	revolute
2	0	0.1	0	$\pi/2$	revolute
3	0	-0.098	0	-$\pi/2$	revolute
4	$\pi/2$	0	0	0	revolute
5	0	0.105	-0.015	$\pi/2$	revolute
6	0	0	0	-$\pi/2$	revolute

Table 3.4: DH-parameters for the NAO right-arm

Base transformation matrix (transformation matrix from first-joint (torso-tuning joint) frame to the base frame):

$$M\_base = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Tool transformation matrix (transformation matrix from end-effector frame to the 6th-joint (RWristYaw,[Kof12]) frame):

$$M\_tool = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & -0.1137 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Then generate the model with *SerialLink* function in Robotics toolbox for MATLAB: figure3.20

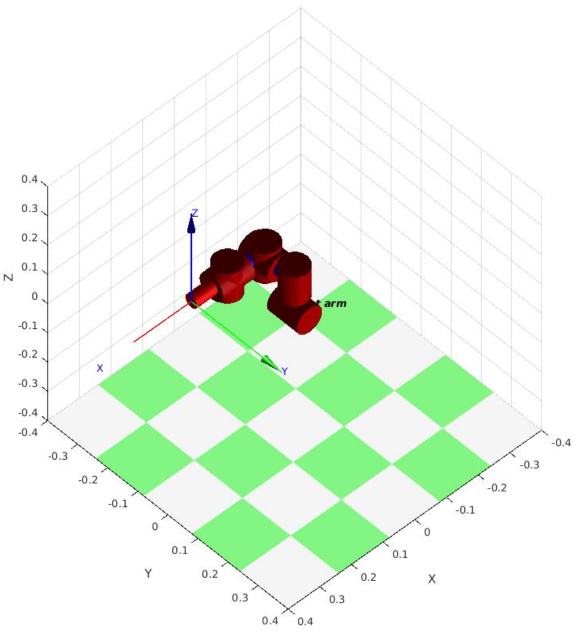


Figure 3.20: NAO right-arm Model

NAO two-arm model

Combine two arm model get the two-arm model, see 3.21. In the individual models, each arm has 6 DoF, 12 DoF in total. In the two-arm model, the first two joints (torso-turning joint and bending joint) of right-arm will only duplicate the motion of the first two joints of left-arm. So each arm has 4 independent DoF and 2 shared DoF, 10 DoF in total.

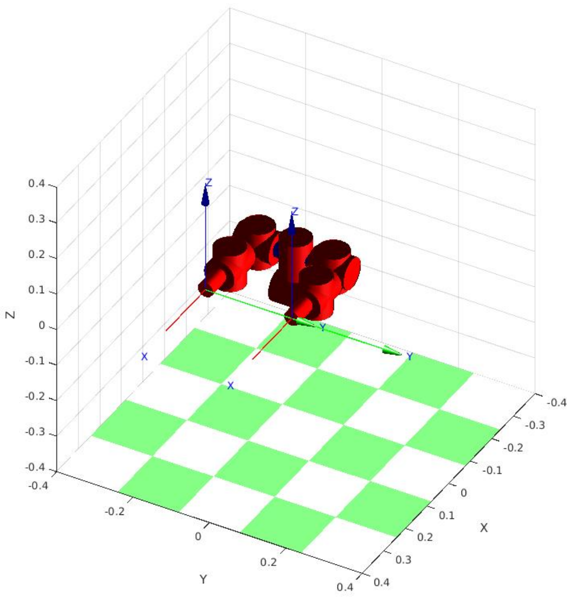


Figure 3.21: NAO two-arm Model

3.2.2 Experimental results

Task definition

Task one: left-arm starts from [0.2187 0.113 0.1] (zero position) to reach [-0.0293 0.2254 0.2783];
Task two:left-arm starts from [0.2187 -0.113 0.1] (zero position) to reach [0.0635 -0.3107 0.0100];
Teaching: no teaching;
Parameter to be minimized: reaching error (distance between end-effector and targeting point).

Colour mapping in the following trajectory figures:
Red: original original trajectories calculated from *minimum jerk trajectory*;
Green: final trajectories after optimization process;
cyan star: targeting points.

Under this task setting, the robot is not possible to reach two targeting points at the same time, because of the limits at the shared joints.

Simulate TTC

Similar to 3.1.2, the first test is to configure the OTC to simulate TTC.
Configuration:
1, set the reaching error of left-arm as B-level constraint, no A-level constraint.

Result 7:
Reaching error leftsrm 0.00448 against 0.1915;
See figure 3.22 3.23 3.24

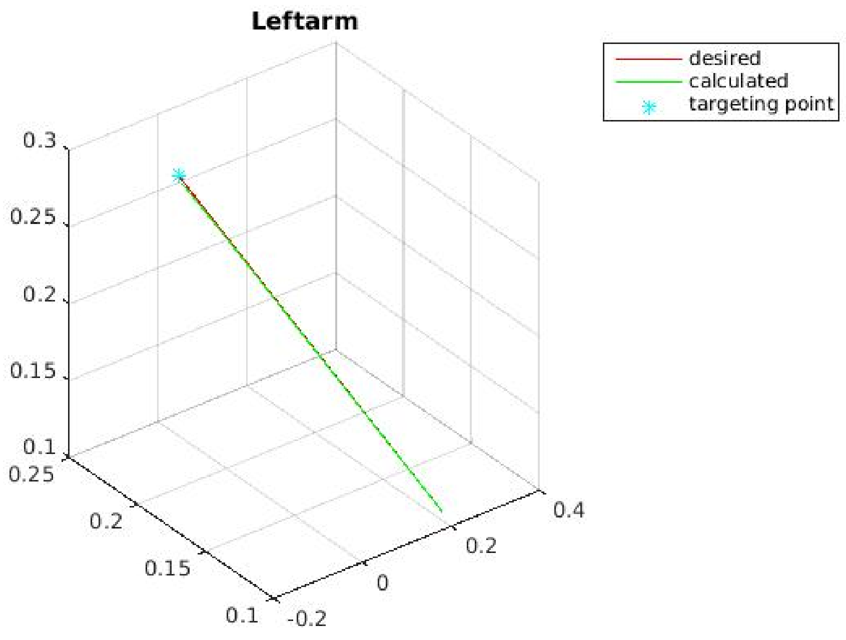


Figure 3.22: Left-arm performance in TTC model

Analysis 7:
The OTC tries the best to drive the left-arm to its targeting point, and the task of right-arm is executed in the null space of left-arm.

Configuration:
1, set the reaching error of right-arm as B-level constraint, no A-level constraint.
Result 8:
Reaching error leftsrm 0.4587 against 0.004338;

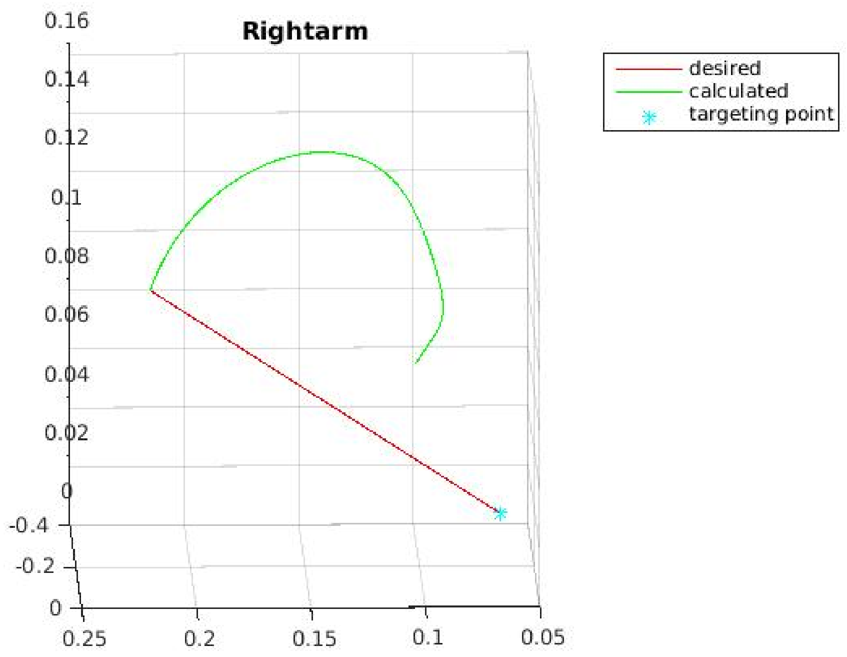


Figure 3.23: Right-arm performance in TTC model

See figure 3.25 3.26 3.27

Analysis 8:
The OTC tries the best to drive the right-arm to its targeting point, and the task of left-arm is executed in the null space of right-arm.

Optimization on both tasks without difference

In this part of test, the OTC will equally drive two arms to reach their targeting points at the same time.
Configuration:
1, set the reaching error of both arms as B-level constraint, no A-level constraint.
Result 9:
Reaching error leftsrm 0.1218 against 0.1218;
See figure 3.28 3.29 3.30

Analysis 9:
Since the robot can not reach two targeting points at the same time and the two tasks must be treated equally, a compromising solution is found to balance the reaching errors of both arms.

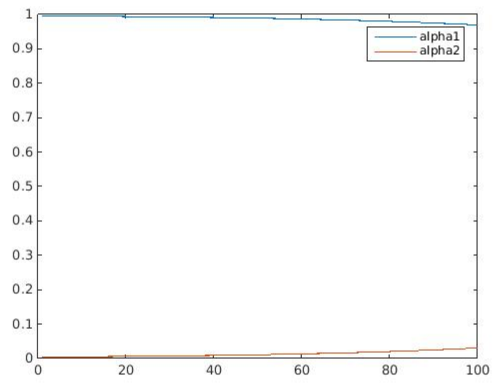


Figure 3.24: Priorities performance in TTC model

Optimization on both tasks with difference

In this part, a tolerance is added to either of the targeting point, as long as the distance between the targeting point and the end-effector is within the tolerance, will be considered as reached. Then the null space is larger, the other arm would get closer to its targeting point.

Configuration:
1, set the tolerance of left-arm target as 0.05;
2, set the left-arm reaching error as A-level constraint, right-arm error as B-level.

Result 10:
Reaching error left-arm 0.0499 against 0.1598;
See figure 3.31 3.32 3.33

Analysis 10:
The OTC drives the arms to reach their targets, and stops moving left-arm closer when the distance is with tolerance; then tries best to move right-arm to get as close as possible to its target. Compare result 10 with result 7, left-arm reaching error is a little bigger in result 10, but right-arm error is smaller.

On the other way round, set tolerance on the right-arm target, gets similar result.

Configuration:
1, set the tolerance of right-arm target as 0.05;
2, set the right-arm reaching error as A-level constraint, left-arm error as B-level.

Result 10:
Reaching error left-arm 0.2608 against 0.0499;

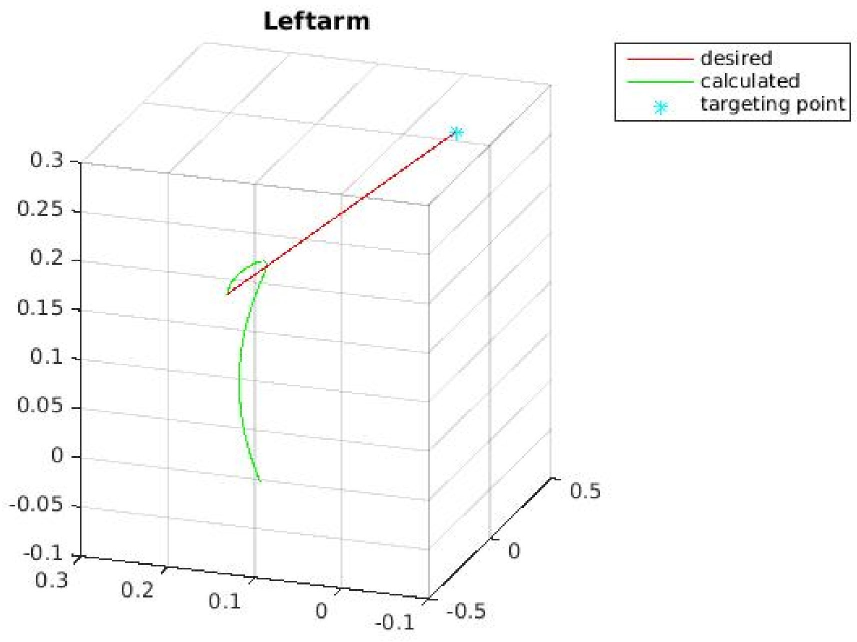


Figure 3.25: Left-arm performance in TTC model

See figure 3.34 3.35 3.36

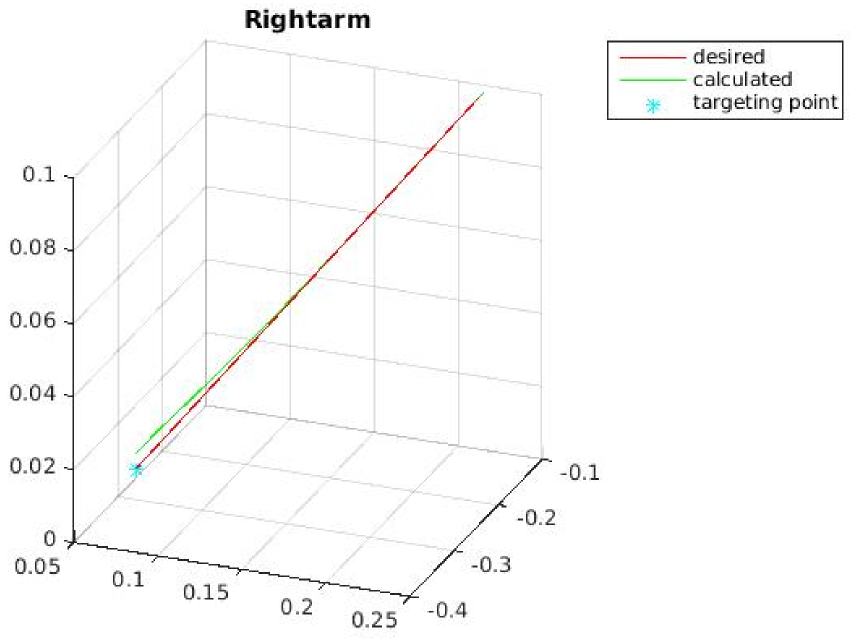


Figure 3.26: Right-arm performance in TTC model

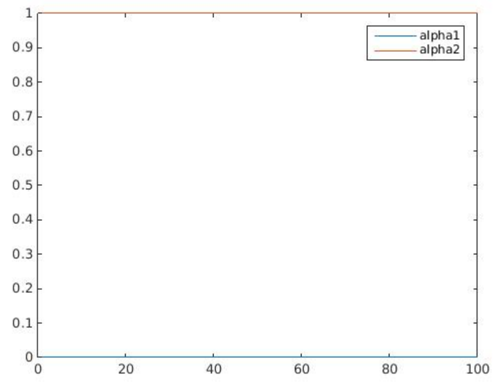


Figure 3.27: Priorities performance in TTC model

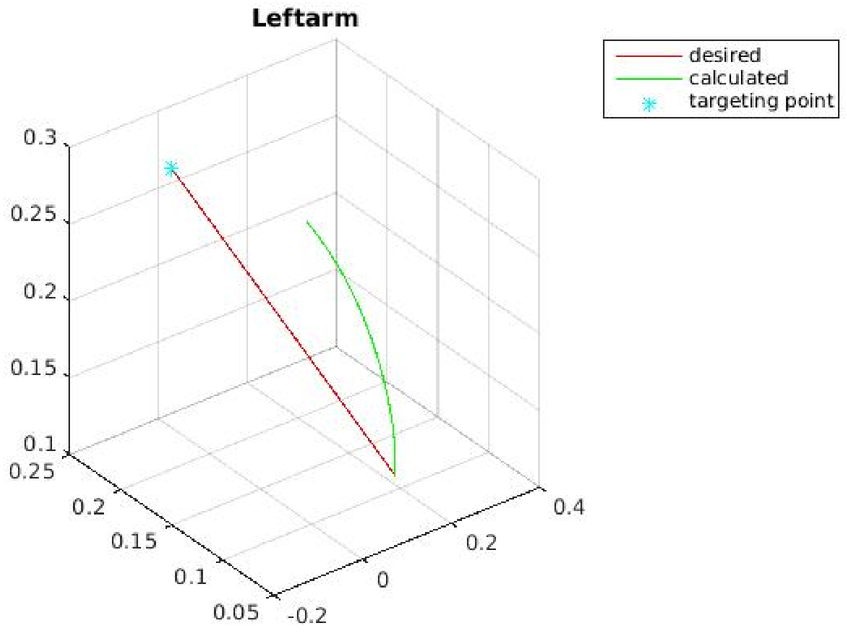


Figure 3.28: Left-arm performance in optimization without difference

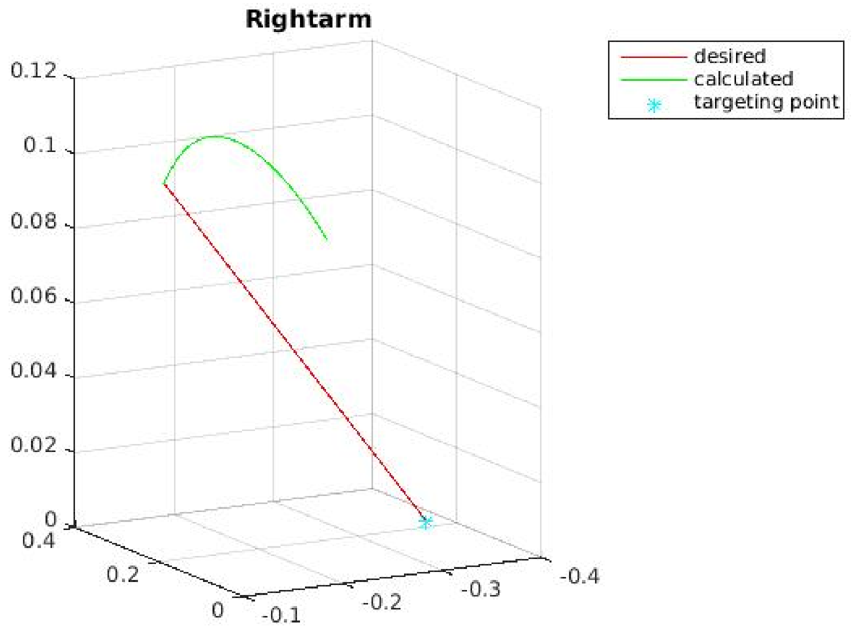


Figure 3.29: Right-arm performance in optimization without difference

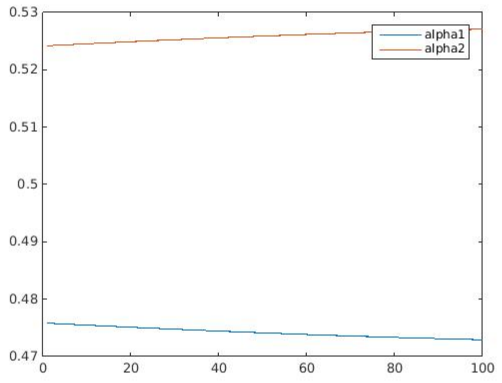


Figure 3.30: Priorities performance in optimization without difference

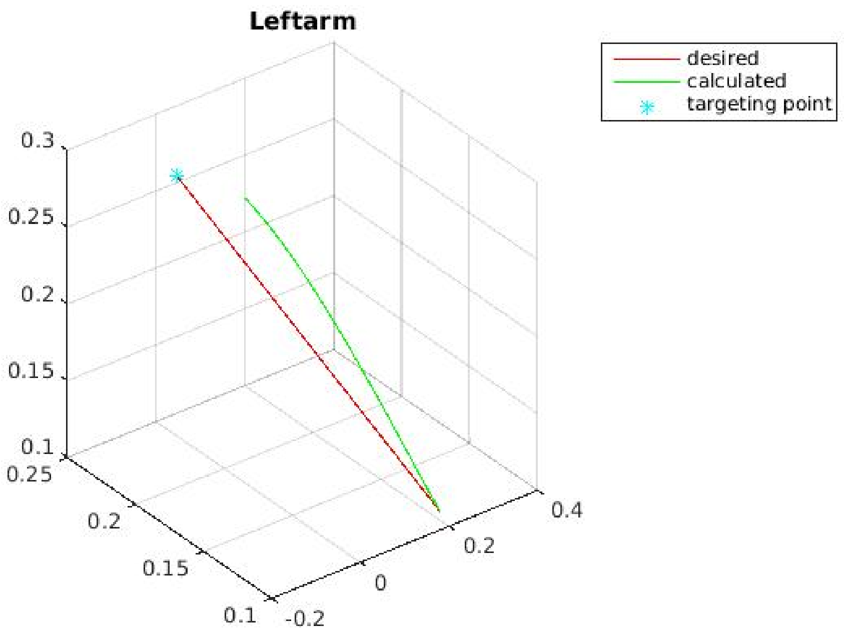


Figure 3.31: Left-arm performance in optimization with difference

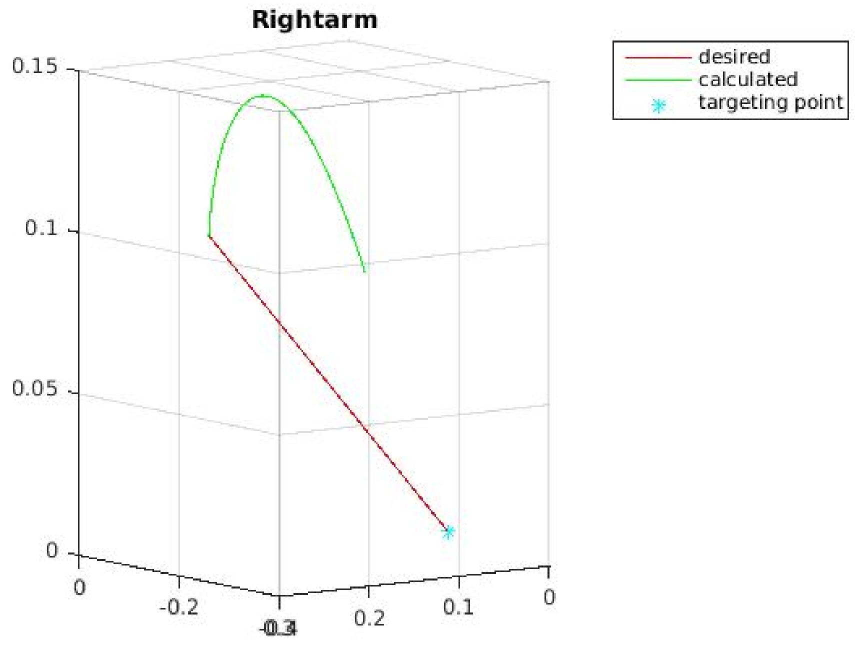


Figure 3.32: Right-arm performance in optimization with difference

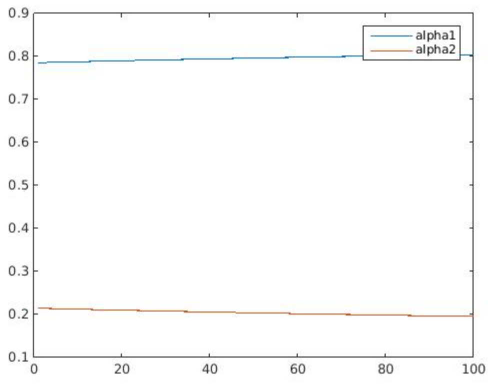


Figure 3.33: Priorities performance in optimization with difference

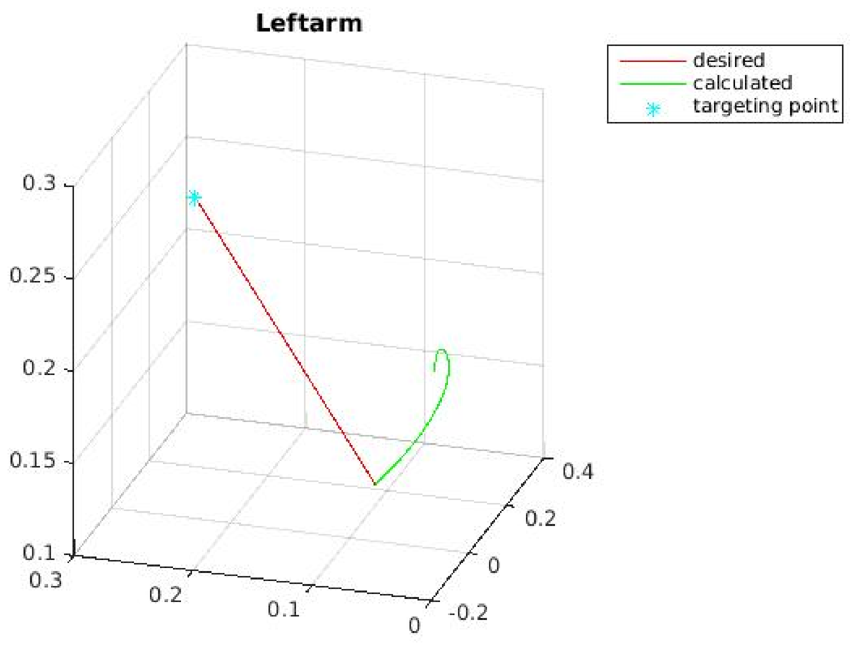


Figure 3.34: Left-arm performance in optimization with difference

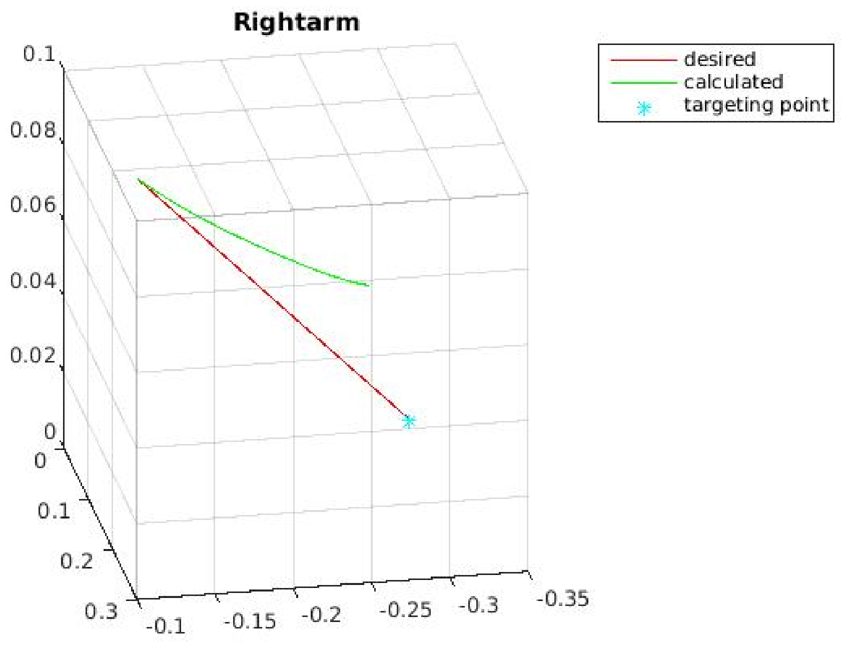


Figure 3.35: Right-arm performance in optimization with difference

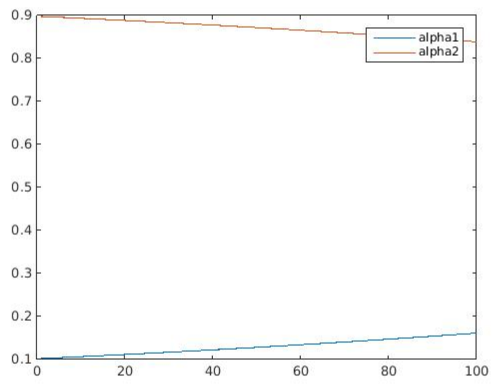


Figure 3.36: Priorities performance in optimization with difference

Chapter 4

Conclusion and future work

This work proposes an approach to optimize the multi-task execution. Each task is assigned a priority which is optimized though (1+1)CMAES algorithm. Key part is the OTC controller which generates the optimized controller to execute the tasks. It can be configured to work in different modes to satisfy different requirements. The approach is tested in KUKA and NAO, and the results show that it works as expected: all the tasks could be executed as best as possible.
In the future work there are some aspects need to be improved. For example the smoothness of the optimized trajectory. In the figure 3.14, after the elbow has reached the acceptance region, the OTC starts driving the robot to satisfy other constraints, which causes a sharp turning in the trajectory. So constraints should be set to guarantee the smoothness. Besides, in our approach, we divide the constraints into two levels to cope constraint contradictions, which works only for small number of constraints. More efficient mechanism should be developed to overcame contradictions from larger number of constraints.

List of Figures

Bibliography

[AH10] Dirk V Arnold and Nikolaus Hansen. Active covariance matrix adaptation for the (1+ 1)-cma-es. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 385–392. ACM, 2010.

[AL15] Sang-ik An and Dongheui Lee. Prioritized inverse kinematics with multiple task definitions. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1423–1430. IEEE, 2015.

[Cor11] Peter Corke. Robotics toolbox for matlab, release 9 [software]. *Robotics toolbox for MATLAB*, 2011.

[DLH09] Martin De Lasa and Aaron Hertzmann. Prioritized optimization for task-space control. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5755–5762. IEEE, 2009.

[IHR07] Christian Igel, Nikolaus Hansen, and Stefan Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary computation*, 15(1):1–28, 2007.

[ISH06] Christian Igel, Thorsten Suttorp, and Nikolaus Hansen. A computational efficient covariance matrix update and a (1+ 1)-cma for evolution strategies. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 453–460. ACM, 2006.

[Kof12] Nikolaos Kofinas. *Forward and inverse kinematics for the NAO humanoid robot*. PhD thesis, Technical University of Crete, Greece, 2012.

[MNR$^+$16] Valerio Modugno, Gerard Neumann, Elmar Rueckert, Giuseppe Oriolo, Jan Peters, and Serena Ivaldi. Learning soft task priorities for control of redundant robots. In *IEEE International Conference on Robotics and Automation (ICRA 2016)*, 2016.

[ODAS15] Christian Ott, Alexander Dietrich, and Alin Albu-Schäffer. Prioritized multi-task compliance control of redundant manipulators. *Automatica*, 53:416–423, 2015.

[SAL15] Matteo Saveriano, Sang-ik An, and Dongheui Lee. Incremental kinesthetic teaching of end-effector and null-space motion primitives. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3570–3575. IEEE, 2015.

[SPB11] Joseph Salini, Vincent Padois, and Philippe Bidaud. Synthesis of complex humanoid whole-body behavior: a focus on sequencing and tasks transitions. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1283–1290. IEEE, 2011.

[SRK^{+}13] Layale Saab, Oscar E Ramos, François Keith, Nicolas Mansard, Philippe Soueres, and Jean-Yves Fourquet. Dynamic whole body motion generation under rigid contacts and other unilateral constraints. *IEEE Transactions on Robotics*, 29(2):346–362, 2013.

[SSVO10] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control.* Springer Science & Business Media, 2010.

License